ALFAMA CURSOS

alfamacursos.com.br

Alfama Cursos

Antônio Garcez
Fábio Garcez
Diretores Geral

Antônio Álvaro de Carvalho
Diretor Acadêmico

MATERIAL DIDÁTICO

Produção Técnica e Acadêmica

Marcela Menezes Flores
Coordenadora Geral

Patrícia Queiroz de Meneses
Coordenadora Pedagógica

José Alves Correia Neto
Renata Jacomo Viana
Autoria

Gabriella Caroline Teles Silva
Sabina Regina Conceição Santos
Revisão Textual

Rafael Rezende de Farias
Editoração

Todos os direitos reservados e protegidos pela Lei 9.610 de 19/02/98.
É proibida a reprodução total ou parcial, por quaisquer meios, sem autorização prévia, por escrito, da ALFAMA CURSOS.

alfamacursos.com.br

Qualidade de Sofware

Apresentação do Curso

Olá caro aluno!

Seja bem-vindo ao curso de Qualidade de Software, sabendo que é uma disciplina de extrema importância para o seu conhecimento. Esta apostila vem complementar as videoaulas.

Lembre-se que apenas pela prática será possível ser um bom programador.

Apresentação do Professor

FÁBIO GOMES ROCHA é formado em Sistemas da Informação e pós-graduado em Engenharia de Sistemas. Atualmente é membro da ABNT (Associação Brasileira de Normas Técnicas), fazendo parte do Comitê 21 - Engenharia de Software e do Comitê 27 - Segurança da Informação. Leciona Técnicas de Desenvolvimento de Sistemas e Linguagem de Programação e na Educação a Distância já trabalhou como desenvolvedor, professor autor e tutor em cursos universitários e técnicos, além de possuir publicações nesta área desde 2006. Além disso, atua na área de Tecnologia da Informação e Comunicação com Desenvolvimento de Sistemas há 17 anos.

Em 2004 publicou um livro sobre Tecnologia para Gestão, intitulado "Secretariado: do Escriba ao Web Writer".

Componente Curricular

EMENTA:

Introdução à Qualidade; Sistema de Gestão da Qualidade; Planejamento de Teste; Teste Unitário; Teste Unitário no PHP; Teste de Performance; Teste de Usabilidade; Planejamento de Avaliação; Métodos de Inspeção; Métodos de Teste de Usabilidade; Métricas de Qualidade; TMMI e CMMI.

PÚBLICO-ALVO:

Alunos oriundos do Ensino Médio e universitários que queiram complementar sua formação profissional.

OBJETIVOS GERAIS:

- Discorrer sobre a garantia de qualidade no desenvolvimento de Software.
- Citar os fatores de garantia de qualidade.
- Aplicar técnicas de revisão, inspeção e garantia de qualidade.

HABILIDADES E ATITUDES ESPERADAS:

- Aplicar conceitos de qualidade no desenvolvimento de Softwares.
- Demonstrar atividades de garantia de qualidade.

Índice

Capítulo 1 - Introdução à Qualidade 8
1.1 - Introdução 8
1.2 - Qualidade de Software 8
Capítulo 2 - Sistemas de Gestão da Qualidade 11
2.1 - Introdução 11
Capítulo 3 - Planejamento de Teste 13
3.1 - Introdução 13
3.2 - Planejtamento de Teste 14
Capítulo 4 - Teste Unitário 15
4.1 - Introdução 15
4.2 - Funcionamento dos Testes Unitários 15
Capítulo 5 - Teste Unitário no PHP 18
5.1 - Introdução 18
5.2 - Instalação 18
Capítulo 6 - Teste de Performance 22
6.1 - Introdução 22
6.2 - Tipos de Teste de Performance 22
6.3 - Fases do Projeto de Teste de Performance 23
6.4 - Plano de Teste de Desempenho e suas Baselines 23
6.5 - Jmeter para Teste de Performance 24
Capítulo 7 - Teste de Usabilidade 26
7.1 - Introdução 26
7.2 - Definindo as Métricas 26
Capítulo 8 - Planejamento de Avaliação 29
8.1 - Introdução 29
Capítulo 9 - Métodos de Inspeção 31
9.1 - Introdução 31
Capítulo 10 - Métodos de Teste de Usabilidade 33
10.1 - Introdução 33
Capítulo 11 - Métricas de Qualidade 35
11.1 - Introdução 35
11.2 - Uso de Métricas 36
11.3 - Razões para se Medir o Software 36
Capítulo 12 - TMMI e CMMI 39
12.1 - Introdução ao CMMI 39
12.2 - Introdução ao TMMI 40
Referências 42

1 – INTRODUÇÃO À QUALIDADE

1.1 – INTRODUÇÃO

São muitas as definições para o termo "Qualidade", porém todas estão associadas a algum ou vários dos seguintes termos: necessidade, satisfação, conformidade, adequação.

"Qualidade é o conjunto das características de uma entidade que lhe conferem a aptidão para satisfazer necessidades exprimidas e implícitas." (ISO 8402)

Portanto, a qualidade depende dos requisitos, ou seja, das necessidades exprimidas.

- Q3uais funcionalidades e características o produto deve ter?

Os requisitos do produto precisam ser definidos, o que nos remete à qualidade dos mesmos. Se os requisitos forem incorretos ou incompletos, ainda que, o processo de produção tenha um padrão de excelência, não será possível atingir a qualidade.

A definição estabelecida na ISO 8402 cita as necessidades implícitas, que apesar de não estarem descritas nos requisitos, precisam ser satisfeitas. Estas necessidades estão frequentemente associadas aos padrões estabelecidos no mercado. Por exemplo: um cliente solicita um Software e apesar de não explicitar entre os requisitos, espera que o mesmo possa ser operado com o mouse e/ou teclado.

Qualidade é a conformidade do produto com os requisitos e os padrões estabelecidos.

Não podemos afirmar que um determinado produto possui um grau maior ou menor de qualidade se o mesmo nunca foi avaliado. A existência da qualidade é comprovada a partir de avaliação. A avaliação baseia-se nas informações coletadas sobre o objeto a ser avaliado, ou seja, em métricas.

A partir da medição das funcionalidades e características previstas de um produto ou serviço é possível avaliar sua qualidade.

A qualidade do produto está associada à qualidade dos processos necessários para a sua produção. Qualquer item considerado insatisfatório na avaliação remeterá à identificação de alguma falha nos requisitos ou nos processos de produção. Depois de corrigida a falha, o novo produto deve ser avaliado. Inicia-se, assim, a melhoria continuada da qualidade, que deve focar os produtos e os processos.

Portanto, requisitos e métricas são fatores essenciais para o processo de avaliação, obtenção e melhoria da qualidade.

1.2 - QUALIDADE DE SOFTWARE

É importante que se entenda que um dos primeiros passos para o desenvolvimento qualitativo de Software é entender o que se quer produzir, e neste sentido, a Engenharia de Software auxilia o processo de planejamento e desenvolvimento do Software.

A Engenharia de Software possibilitou a criação em menor tempo de novos aplicativos mais complexos e com maior qualidade. O século XXI mostrou com a melhoria das aplicações web e a evolução do conceito da computação na nuvem (cloud computing), onde tudo que é aplicativo ou serviço está disponível de forma remota, em qualquer lugar, o quão rápido o ser humano pode chegar quando os processos são organizados por mais básicos que sejam.

Fica a dica!

No mundo atual, desenvolver um produto sem qualidade leva qualquer empresa à morte "falência" e metodologias vêm sendo criadas para a melhoria da qualidade. Para ficar por dentro das novidades sobre qualidade, acesse os sites:

- www.infoq.com
- www.fgrweb.com.br

Recordando

A qualidade é um conceito antigo, mas em Software, medir a qualidade é algo complexo e difícil, para tanto existem normas e metodologias que facilitam que o desenvolvedor crie produtos de qualidade. A qualidade é essencial para a sobrevivência de produtos.

2 - SISTEMAS DE GESTÃO DA QUALIDADE

2.1 – INTRODUÇÃO

Os consumidores sempre estiveram atentos à qualidade das mercadorias que compravam ou trocavam. Nas primeiras fases, o método utilizado baseava-se na experiência do adquirente, que observava, media, comparava, experimentava, enfim, conferia a qualidade. Quando a mercadoria era um produto, o próprio produtor também fazia uma análise da qualidade do mesmo, buscando melhorá-lo, objetivando satisfazer o cliente e conquistar maior participação no mercado.

Com o desenvolvimento do processo de produção em massa aumentou a importância do controle de qualidade, afinal, uma unidade produzida com defeitos pode comprometer a credibilidade de toda a produção e até do produtor. Surgiram diversas ferramentas e técnicas visando ao controle e ao aumento da qualidade. Os resultados positivos conquistados levaram ao desenvolvimento de sistemas mais complexos que, além de controlar a qualidade após a produção, abordavam o ciclo de produção e suas implicações. Estes sistemas foram os primeiros Sistemas de Gestão da Qualidade - SGQ.

O reconhecimento dos direitos dos consumidores levou à criação de diversas regulamentações, que definem requisitos específicos para que um produto possa ser comercializado ou para que o mesmo obtenha determinados certificados de qualidade. O SGQ se desenvolveu visando, além da qualidade, atender aos requisitos destas regulamentações.

A base para o desenvolvimento dos modelos de SGQ é o ciclo Plan, Do, Check, Act – PDCA, inspirado na metodologia científica de Francis Bacon e popularizado por W. Edwards Deming.

O ciclo de Deming apresentado a seguir, baseia-se em planejar, executar, checar e agir.

Figura 01: Ciclo de Deming.

Atualmente, existem diversos modelos de SGQ, alguns genéricos e outros para áreas específicas. Algumas instituições optam por desenvolver seu próprio SGQ e outras adotam sistemas desenvolvidos por instituições especializadas.

Os SGQs podem focar processos ou projetos e alguns possuem estrutura para controlar a qualidade de ambos.

"Um processo é uma série de ações que geram um resultado." Guia PMBOK, 4ª Ed.

Processo é um termo derivado de "processu", em latim, que significa proceder, avançar.
"Um projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo." Guia PMBOK, 4ª Ed.

Projeto é um termo derivado de "projectu", em latim, que significa "lançar para adiante".

Ao implementar um Sistema de Gestão da Qualidade pode-se optar por:

- Desenvolver e implementar um sistema próprio.
- Adotar e implementar um sistema de mercado.
- Adaptar um ou mais modelos para formatar e implementar um sistema próprio.

Ainda que a opção seja desenvolver um sistema próprio, é necessário um estudo sobre os sistemas já existentes, sem o qual se expõe aos riscos de cometer erros conhecidos, utilizar recursos para desenvolver soluções já existentes, entre outros.
Para subsidiar a implementação de um sistema de qualidade, dois instrumentos são de fundamental importância:

- **Modelos de Gestão (Frameworks):** Metodologias para aplicação de boas práticas de governança corporativa e/ou de gestão de projetos.
- **Normas:** Documentos que estabelecem regras, diretrizes e exigências.

Obs.: De forma geral, as normas definem enquanto os modelos sugerem o que deve ser feito.

Existem muitos modelos voltados para o desenvolvimento de Software, alguns associados às linguagens de programação, outros genéricos. Também existem várias normas que abrangem o processo de Software. As novas versões dos modelos e das normas devem ser avaliadas regularmente, objetivando-se o aproveitamento das melhores práticas da atualidade e da adequação às exigências das normas.

Independentemente do modelo utilizado, a atividade de desenvolvimento de Software deve ser tratada como um projeto. O capítulo específico de projetos abordará em detalhes a utilização de metodologias voltadas ao gerenciamento de projetos.

3 - PLANEJAMENTO DE TESTE

3.1 – INTRODUÇÃO

A fase de testes de Software é mais uma área de conhecimento que se compromete com a qualidade. Testar é a atividade de analisar e executar aplicações com a intenção de verificar se o comportamento dos sistemas é o esperado. O objetivo dos testes de software vai bem além de revelar defeitos. A literatura traduz um defeito (ou bug) como um passo, processo ou definição de dados incorretos. Um defeito poderá causar um erro (um estado inconsistente), que resultará em falhas (comportamentos que diferem do esperado). Diferente de uma revisão simples, ou análise estática de código, o teste compreende vários mecanismos baseados nos conceitos de Engenharia de Software. Um caso de teste é composto por dados de entrada e resultados esperados. Para um ou mais casos de testes há um oráculo, um mecanismo que decide sobre a correção de uma execução. Porém, há oráculos que são complicados, devido ao desconhecimento dos resultados, ou resultados esperados difíceis de ser avaliados ou até mesmo devido à grande quantidade de dados. Para resolver este e alguns dos problemas frequentes na definição de estratégias de testes, há critérios de testes que envolvem uma seleção de um domínio e elaboração de caso de testes.

Portanto, a etapa de testes é um processo de verificar e validar o que em uma aplicação de Software atende às regras técnicas e de negócio que conduzem ao projeto em implementação.

A Qualidade de Software como visto anteriormente passa por várias etapas, entre elas o Teste de Software, mas esse teste é algo que começou há pouco tempo. Para ter uma ideia, o primeiro bug surgiu nos primórdios da computação, aproximadamente em 1947. Inicialmente as falhas eram de Hardware, mas com o passar do tempo o Hardware reduziu os erros e o Software passou a ser o problema.

O conceito de Teste de Software é algo relativamente novo. Em 1957 foi estabelecida uma distinção inicial entre tirar os defeitos e testar o Software por Charles L. Baker. Em 1961 o livro Computer Programming Fundamentals de Leeds e Weinberg foi o primeiro a apresentar um capítulo sobre Teste de Software, o que representou uma inovação no conceito de Qualidade de Software.

Depois, em 1967 foi publicado um trabalho por William Elmendorf da IBM tratando sobre abordagem para testes funcionais controlados, algo ainda novo na época, seguido em outubro de 1968 pelo relatório publicado na conferência sobre Engenharia de Software patrocinada pela OTAN, uma das primeiras a tratar sobre a garantia de Qualidade de Software.

Depois, em 1969 na conferência para o Comitê de Ciência da OTAN, na Itália, Dijkstra foi o primeiro a afirmar que o teste deve mostrar a presença e não a ausência de defeitos.

Já em 1970, Wiston Royce publicou o artigo na IEEE sobre o modelo de desenvolvimento em cascata. Em 1973, Willian Hetzel apoiado pela Universidade da Carolina do Norte publicou o livro "Métodos de Teste de Programas". No mesmo ano William Elmendorf desenvolve o gráfico de causa e efeito.

Em 1976, Michael Fagan publicou um trabalho pela IBM que propõe a inspeção de códigos, visando à redução de retrabalhos. No mesmo ano Thomas McCabe introduziu o conceito de complexidade ciclomática como uma das métricas de Software em um artigo publicado pelo IEEE. Enfim, Myers publica um dos primeiros livros sobre Confiabilidade de Software tratando dos princípios e práticas para criação de Softwares confiáveis. Neste livro, Myers afirma que o objetivo do testador é fazer o programa falhar.

Já em 1977 Jim McCall, Pall Richards e Gene Walters fizeram um trabalho para a força aérea americana listando 55 fatores para a Qualidade de Software. No mesmo ano, Tom

Gilbs publica o livro "Métrica de Software", que foi e é considerado o livro base para diversas métricas de Software.

Por fim, em 1979 Myers publicou o primeiro livro tratando apenas de Teste de Software, intitulado "A arte de testar software". É ainda hoje considerado por muitos o livro base para Teste de Software. Em 1981, Barry Boehn publica o livro "Software Engeering Economics", introduzindo o conceito de que o custo de um defeito cresce exponencialmente no tempo.

Desta forma, o Teste de Software é de fundamental importância para avaliar se o Software atende aos requisitos funcionais e não funcionais, ou seja, se está dentro do padrão de qualidade previsto. O objetivo principal do processo de teste conforme vários autores é encontrar falhas, ou seja, não é verificar se a aplicação está funcionando, e sim buscar o maior número de erros possíveis.

Apesar dos testes normalmente estarem relegados ao segundo plano, ou nem serem executados, é um processo que deve acompanhar o desenvolvimento do início até a conclusão do produto. Os testes caixa branca como testes unitários, por exemplo, devem ser feitos durante o processo de desenvolvimento. Os testes de caixa preta como os de funcionalidade, por exemplo, buscam falhar nos requisitos funcionais no produto funcionando, e já os testes de desempenho, que estão ligados aos requisitos não funcionais, buscam verificar se o sistema está atendendo requisitos de recursos, velocidade adequada, suporte e etc.

3.2 - PLANEJAMENTO DE TESTE

O planejamento de teste é parte crucial para o sucesso de um teste bem executado, para tanto é necessário criar um plano de execução de testes.

O plano de teste é desenvolvido com base na documentação do sistema, a exemplo temos:

- **Plano de Teste Funcional:** deve testar as funcionalidades que seguem a base do documento caso de uso. Com base neste, é criado um caso de teste (com o mesmo formato). Na verdade, utilizaremos o caso de uso para ver se as funcionalidades descritas no caso de uso estão presentes no sistema.
- **Teste de Performance:** utilizamos os requisitos não funcionais como: quantos usuários devem utilizar simultaneamente o sistema, qual o desempenho aceitável e em qual tipo de equipamento este sistema vai rodar. Todos os testes são criados com base na documentação, por este motivo que sistemas mal documentados, normalmente são sistemas mal testados.
- **Teste Unitário:** é criado em conjunto com a codificação, este necessita do diagrama de classe para poder ser criado. Em técnicas como TDD, primeiro criamos o teste para depois codificarmos o sistema.

4 - TESTE UNITÁRIO

4.1 – INTRODUÇÃO

Sabemos que errar é humano e, mesmo bons programadores cometem erros, a diferença é: Quando estes erros serão detectados? O bom programador utiliza recursos para detectar o erro o quanto antes, pois como vimos, erros encontrados no início do desenvolvimento são mais baratos do que detectados ao final do processo de desenvolvimento. O teste unitário é parte do processo de desenvolvimento, serve para verificar a compatibilidade e o funcionamento de uma unidade do sistema.

O teste unitário é um teste de caixa branca, pois precisa ter acesso ao código para ser feito. Ele normalmente serve para testar as classes ou funções do sistema.

O objetivo é garantir a qualidade dos componentes, lembrando da importância da qualidade para a produção de Software. Isto vai garantir que o componente funcione como foi planejado. Os testes unitários utilizam os xUnits, que são técnicas que buscam viabilizar os testes em diversas plataformas e linguagens de programação. Vamos às técnicas de teste unitário com o PHPUnit e com o JUnit, mas a maioria dos testes unitários como o CUnit é idêntica, pois seguem um padrão.

4.2 - FUNCIONAMENTO DOS TESTES UNITÁRIOS

O teste unitário serve para verificar se o programa se comporta como o esperado, realizando baterias de testes com base no caso de teste já programado. Os testes validam partes, ou seja, fragmentos do código, o que será chamado aqui de unidade. O teste unitário, então, testa a unidade do código e o mesmo pode ser feito para criar sistemas com base nos testes também, ou seja, primeiro criamos os testes para depois criarmos a unidade.

O termo "Teste Unitário" é devido ao fato de os testes serem realizados em apenas uma parte, ou unidade, podendo ser utilizado para testar uma função, uma classe e etc.

Qual o objetivo do teste unitário?

O principal objetivo de criar e realizar testes unitários em códigos é verificar se a classe ou função se comporta como desejado, evitando erros futuros e garantindo a funcionalidade do sistema.

Além disso, a utilização de Frameworks de testes como o PHPUnit, torna o processo automatizado e não interfere no código.

Em métodos ágeis como o Scrum e o XP, por exemplo, os testes unitários fazem parte da metodologia de desenvolvimento, sendo indicado iniciar o processo pelos testes, ou como deve ser chamado, desenvolvimento orientado a teste, pois o teste vai servir como documentação do sistema, evitando excesso de documentos e garantindo a qualidade das funcionalidades.

Entre as principais vantagens de se utilizar uma metodologia com base em teste unitário está:

- Assegurar a qualidade de cada trecho de código isolado.
- Uma vez que o código do teste foi escrito, o mesmo poderá ser repetido por diversas vezes.
- É uma forma automatizada de realizar testes.
- De certa forma documenta (exemplifica) o uso do trecho que está sendo testado.

- Não interfere no código fonte de sua aplicação.

Mas é óbvio que também há desvantagens:

- Não existe milagre, você tem que escrever o código do teste.
- Não consegue testar a aplicação por completo ou grandes contextos com um único teste.

Fica a dica!

Em diversas metodologias, o teste unitário é extremamente importante, entre elas temos o RUP e para saber mais visite o site abaixo:

- http://www.wthreex.com/rup/process/modguide/md_untst.htm

Recordando

O teste unitário tem como principal objetivo testar partes do código e verificar se há garantia de qualidade.

5 - TESTE UNITÁRIO NO PHP

5.1 - INTRODUÇÃO

Seguindo os padrões de xUnits, apresentaremos aqui o PHPUnit. O primeiro passo em qualquer forma de teste unitário é a instalação do sistema de unidade de testes, e dependendo do caso, integrar com seu ambiente de desenvolvimento (IDE).

5.2 - INSTALAÇÃO

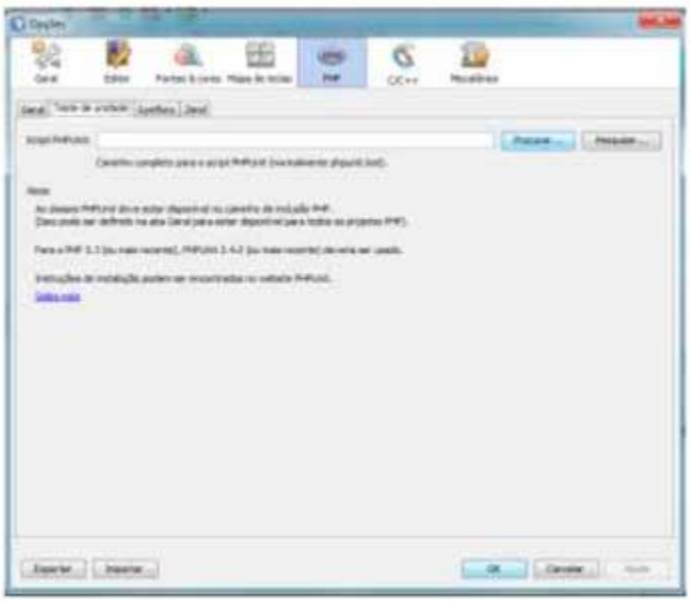
Ps.: A versão PHPUnit 3.5 requer a versão do PHP 5.2.7
Para instalar, vamos utilizar o sistema PEAR de instalação de pacotes.
Primeiramente, vamos instalar os canais PEAR utilizados pelo PHPUnit.

pear channel-discover pear.phpunit.de
pear channel-discover components.ez.no
pear channel-discover pear.symfony-project.com

Isto é feito apenas uma vez para poder efetuar a instalação do PHPUnit.
Agora, devemos instalar o PHPUnit.

pear install phpunit/PHPUnit

Para utilizar o PHPUnit no NetBeans faça:
Menu Ferramentas --> Opções --> PHP --> Teste de Unidade.
No campo Script PHPUnit clique em buscar e selecione o arquivo phpunit.bat.
Pronto, o NetBeans está preparado para criar testes unitários.



Depois de instalado o sistema de Unit, é possível criar os testes unitários e efetuá-los. Para começar, será feita uma demonstração de teste unitário e seguiremos para uma explicação completa dos recursos dos testes unitários.

Os testes unitários como comentados anteriormente, devem testar uma função ou uma classe como demonstração. Vamos criar uma classe Cálculo, que serve para fazer soma e

subtração em PHP para demonstração.

```
class Calculo
{
   public function somar($a, $b)
   {
      return $a+$b;
   }

   public function subtrair($a, $b)
   {
      return $a-$b;
   }
}
```

Agora vamos criar o teste que utilizará os recursos do PHPUnit para verificar se a classe que criamos está funcionando de forma adequada.

```
<?php
// Especifique o caminho do PHPUnit
require_once '/usr/share/php/PHPUnit/Framework.php';
// Especifique o caminho da classe a ser testada
require_once 'calculo.php';
/* Como a classe a ser testada se chama "calculo", a classe de teste deverá se chamar "CalculoTest" e (neste caso) irá herdar de PHPUnit_Framework_TestCase
*/
class CalculoTest extends PHPUnit_Framework_TestCase
{
      // Funções de teste devem ter o prefixo "test"
      public function testSoma()
      {
            // A classe a ser testada é instanciada
            $varcalc = new Calculo();
             // Testamos a afirmação que "4" é o resultado de passar "2" e "2" (ou seja, 2 + 2 =4)
            $this->assertEquals('4', $varcalc ->somar(2, 2));
      }
      public function testSubtracao()
      {
            // idem (veja função acima)
            $ varcalc = new Calculo();
            $this->assertEquals('0', $varcalc ->subtrair(2, 2));
      }
}
?>
```

Para analisar se os testes estão funcionando, clique com o botão direito sobre o item e mande o NetBeans testar. Ele vai exibir se está funcionando ou não. Para verificar um erro, você pode alterar dizendo que 2 – 2 são 3 e ver o resultado.

Outra forma de executar é pela linha de comando, utilizando o comando PHPUnit Calculoteste.php.

O resultado seria semelhante ao abaixo:

```
PHPUnit 3.3.16 by Sebastian Bergmann.

.

Time: 0 seconds
```

OK (2 tests, 2 assertions)

Como funcionou?

O Framework, o xUnit, o PHPUnit, em conjunto com a classe que criamos chamada CalculoTest efetuaram o teste. A classe CalculoTeste herdou as características do PHPUnit e pode utilizar a função assertEquals, que tem como objetivo analisar se o resultado é igual, caso seja diferente ele retorna um erro.

Aparentemente é mais complicado, já que o código do teste é extenso e também demora muito tempo criando os testes, mas lembre-se que a qualidade é garantida com os testes, além do mais, agora você pode reaproveitar o código gerado sabendo que é totalmente funcional.

Além disso, a prática torna o processo mais rápido. A estrutura básica que foi digitada pode ser gerada também pelo NetBeans, clicando em Novo --> Teste. Ele gera parte do código de teste, tornando o processo mais simples e fácil.

Fica a dica!

Os testes unitários são a base para o TDD, XP, entre outros. É necessária a prática constante para poder utilizar as técnicas de TDD. Visite os sites abaixo para manter-se atualizado e obter informações:

- http://www.phpunit.de/manual/3.4/en/index.html
- www.fgrweb.com.br/wp

O SimpleTeste é outra ferramenta de teste unitário mais simples, pois não precisa ser instalado. Para obter informações acesse: simpletest.org/.

Recordando

O PHPUnit é um Framework para teste unitário, seguindo o padrão do xUnits do mercado, que além de possibilitar a criação de testes unitários ajudam na garantia de qualidade das classes e funções.

6 - TESTE DE PERFORMANCE

6.1 – INTRODUÇÃO

O teste de performance tem por objetivo buscar falhas de performance no sistema, encontrar os gargalos de velocidade, analisar o suporte a Hardware, a velocidade com uma quantia x de usuários simultâneos para saber os limites do Software, etc.

> Segundo (Bastos et. al., 2007) "o teste de desempenho garante a velocidade de processamento de informação". Ainda segundo Shari Lawrence Pfleeger (2003) "os testes de desempenho têm como base os requisitos, portanto, os tipos de testes são determinados pelos tipos de requisitos não funcionais especificados".

Os testes de desempenho, segundo Pfleeger, dividem-se em teste de estresse, teste de volume, teste de configuração, teste de compatibilidade, teste de regressão, teste de segurança, teste de tempo, teste de ambiente, teste de qualidade, teste de recuperação e teste de manutenção.

Nós preferimos, mesmo utilizando o padrão citado por Pfleerger, utilizar uma divisão mais simples, como teste de ambiente físico, ambiente lógico e do sistema, separando o teste em três etapas distintas: o pré-teste, o teste e o pós-teste, e assim conseguir atingir o objetivo dos testes. O principal objetivo de se fazer um teste de desempenho, é verificar se o sistema realmente vai funcionar no ambiente designado e se vai conseguir manter um desempenho adequado durante um determinado tempo, reduzindo custos operacionais e de equipamentos.

O teste permite estimar as características de desempenho de uma aplicação, bem como prever as necessidades de equipamentos e prever futuros problemas de desempenho, levando a prever recursos para o crescimento futuro da aplicação.

Desta forma, podemos dizer que o teste de desempenho fornece os dados para futuras probabilidades de insatisfação dos usuários com o desempenho do sistema. Fornece ainda os dados para auxiliar na prevenção de perdas de receita ou credibilidade com o sistema, pois a empresa pode prever a escalabilidade e os recursos que garantem a estabilidade do sistema.

Os testes de desempenho também indicam os pontos de tunning do sistema, onde a equipe de desenvolvimento deve atuar para melhorar a escalabilidade, o desempenho e a confiabilidade do sistema. A equipe de teste de desempenho tem um papel importante, pois é quem analisa os componentes críticos do sistema, sendo o facilitador de tunning para o sistema.

6.2 - TIPOS DE TESTE DE PERFORMANCE

Existem diversos tipos de teste de desempenho, os mais comuns são: estresse, volume, regressão, carga e tempo e ambiente. Abaixo segue a descrição de cada um dos testes.

- **Teste de estresse:** avalia o Software quando este é levado aos seus limites por um determinado período, ou seja, quando ele está sob condição crítica, apresentando uma forma de verificar se o Software terá um desempenho quando o sistema estiver utilizando o máximo de usuários simultaneamente no mínimo de recursos computacionais. Assim é possível analisar o pico do uso dos recursos e analisar se o mesmo suportará a demanda, simulando assim um ambiente de produção real e verificando as restrições deste ambiente de operação.

- **Teste de volume:** avalia como o Software se comporta quando grandes quantidades de dados são inseridas, verificando os limites para a sobrecarga do sistema. Além disso, segundo Pfleeger, "verifica os campos, registros e arquivos para ver se seus tamanhos podem acomodar todos os dados esperados, garantindo assim que o sistema reaja de maneira conveniente quando os conjuntos de dados alcançam seu tamanho máximo".
- **Teste de regressão:** é o retorno do teste, ou seja, serve para garantir que o desempenho do novo sistema seja igual ou superior ao sistema anteriormente testado.
- **Teste de segurança:** serve para garantir que os requisitos mínimos de segurança foram atingidos, validando os requisitos de disponibilidade, integridade e confiabilidade conforme a norma ISO/IEC 27002.
- **Teste de carga e tempo:** são testes que verificam o tempo necessário às respostas do sistema, além de verificar os limites operacionais do sistema de acordo com a carga de trabalho. Funciona junto com o teste de estresse normalmente.
- **Teste de ambiente:** serve para avaliar a capacidade real do sistema de funcionar no local de instalação, analisando os requisitos como interrupção de energia, recursos do ambiente, etc.

6.3 - FASES DO PROJETO DE TESTE DE PERFORMANCE

Independente do sistema a ser testado, as fases do projeto continuam as mesmas, o que vai mudar são as atividades do produto, porém todas possuem entradas e saídas a serem analisadas, sendo dividido segundo Leonardo Molinari (2009, p. 65) em "definição do processo, planejamento de testes e cenários, construção de cenários, executando cenários e analisando cenários".

6.4 - PLANO DE TESTE DE DESEMPENHO E SUAS BASELINES

O plano de teste deve conter as baselines para o teste de desempenho. A criação de uma baseline para o teste de desempenho é de fundamental importância para o sucesso dos testes de desempenho de um sistema. Durante a criação da baseline são definidas as métricas de avaliação e eficácia para o teste. É possível analisar se o desempenho está melhorando ou piorando e encontrar os desvios em diferentes ambientes. Como exemplos temos a medição do tempo de carregamento de dados, o uso de recursos como processador, a memória e etc. É possível dividir o teste em diversos itens ou diversas linhas bases, mas para criar o plano de teste devemos ter como base o ambiente em que o sistema foi planejado para funcionar, ou seja, sem ter em mente ambiente usuários máximos seria praticamente impossível fazer um teste e saber se o sistema vai ter um bom funcionamento em um mundo real.

Para a criação do caso de teste são necessários os dados de requisitos não funcionais, caso não tenha tal documentação, será necessário realizar uma entrevista para fazer o levantamento de informações como:

- Ambiente em que o sistema vai funcionar.
- Quantidade de usuários simultâneos no sistema.
- Velocidade da rede.

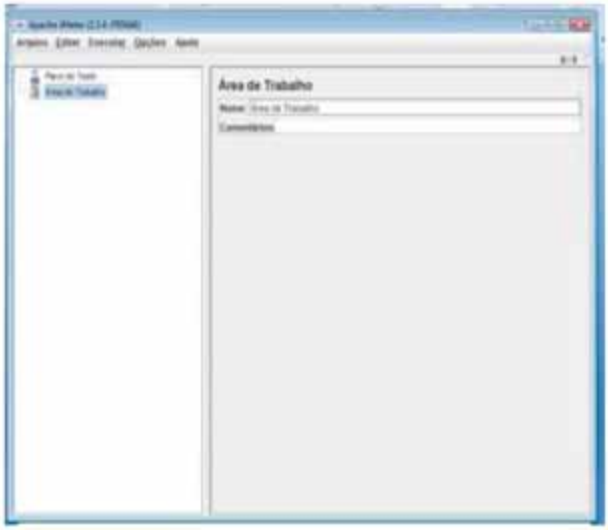
Para que os testes sejam feitos de forma adequada, é recomendada a utilização de um documento de análise de baseline da aplicação, de forma a ter uma métrica adequada. O documento conteria os dados necessários aos testes de forma a nortear o analista no plano de teste.

O teste de Software serve para buscar falhas de concepção, funcionalidade e etc. Os testes de funcionalidade buscam, por exemplo, falhas nos requisitos funcionais, já os testes de desempenho tratam dos requisitos não funcionais, ou seja, os requisitos de recursos, velocidade, suporte e etc.

6.5 - JMETER PARA TESTE DE PERFORMANCE

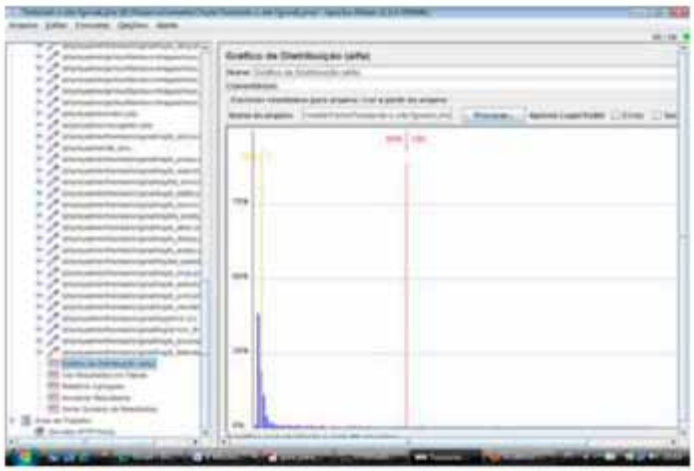
O JMeter é um Software livre que pode ser utilizado para a realização do processo de teste de performance. Este pode ser obtido gratuitamente na internet.

Para utilizar o JMeter é necessário ter instalado o Java. A tela do JMeter possui duas áreas, uma para o planejamento de teste e uma para a área de teste, como pode ser visto na figura abaixo.



Todo o plano do que será realizado deve ser feito no Plano de Teste e para a maioria dos casos, a área de trabalho vai apenas conter um servidor próxy que vai capturar o que o testador deseja testar.

O JMeter tem entre os pontos positivos, um conjunto de relatórios que possibilita a análise dos testes, como pode ser visto na figura abaixo.



Fica a dica!

Os testes de performance são de extrema importância, e uma ferramenta essencial para este tipo de teste é o JMeter, ferramenta gratuita que possibilita a criação de testes de performance. Ela pode ser obtida gratuitamente em: jmeter.apache.org/.

Para ver um passo a passo de como criar e realizar um teste de performance, leia um artigo que está disponível no site: http://fgrweb.com.br/wp/?p=68.

Recordando

O Teste de performance garante a qualidade, em se tratando de requisitos não funcionais, tornando possível saber os limites do sistema.

7 - TESTE DE USABILIDADE

7.1 – INTRODUÇÃO

Há uma competência importante para a sobrevivência de muitos sistemas computacionais. Esta competência, a usabilidade, é considerada como um dos principais fatores que determinam a consolidação de grande parte dos projetos de tecnologia de informação e comunicação.

Usabilidade é a capacidade que um sistema possui de ser entendido, aprendido e utilizado sob condições estabelecidas de operação (ISO/IEC, 9126-1). Também é possível caracterizá-la como a habilidade de um sistema interativo que, em determinado contexto realiza tarefas de maneira eficiente, eficaz e agradável (ISO/IEC, 9241). Enquanto o esforço e o tempo são atributos relacionados com a eficiência da aplicação, a eficácia concentra-se nas tarefas para atingir os reais objetivos do sistema, e quem define a aceitação de um produto, ou seja, se este será bem recebido, é o usuário final que de fato utilizará o sistema.

O usuário deverá ser o elemento central do processo de desenvolvimento de sistemas. Portanto, o desenvolvedor deverá considerar suas habilidades e limitações de uso na aplicação. É por isso que o uso do sistema deverá ser suficientemente intuitivo para a assimilação de seus comandos. Assim é possível afirmar que o aprendizado do usuário é um critério fundamental para usabilidade desde que os sistemas deverão ser fáceis de operar. Para os especialistas em usabilidade qualquer atributo que dificultar a compreensão, resultará fatalmente em rejeição do usuário. É por esta razão que argumenta-se o uso de boas práticas para adquirir-se produtividade. Por esta razão, a usabilidade é considerada como um dos parâmetros competitivos para a qualidade no processo de produção de Software.

A prática de usabilidade é essencial para qualquer tipo de empresa não importando seu porte; desde pequenas e médias empresas até grandes multinacionais. É também indiferente quanto ao setor; seja público, privado ou de capital misto. Porém, o propósito deste tipo de prática é diferente para cada empresa e nem todas dão a atenção necessária em decorrência de prazos e orçamentos curtos. Atualmente, a preocupação com a usabilidade é mais perceptível em empresas de comércio eletrônico que possuem como prioridade a satisfação dos clientes, como garantias de aumento de vendas. Algumas empresas de capital misto tendem a empregar a metodologia centrada no usuário para adquirir maior produtividade de seus colaboradores e reduzir os custos de projetos.

Portanto, empregar usabilidade em sistemas computacionais é realmente um bom negócio. Além de ser um aspecto competitivo de mercado, otimiza o uso de recursos da empresa.

7.2 - DEFININDO AS MÉTRICAS

Os problemas de usabilidade são detectados através de avaliações com ou sem o envolvimento de usuários finais. É essencial que antes de iniciar as avaliações sejam estabelecidas as métricas para determinar se um projeto está cumprindo seus objetivos.

As métricas de usabilidade são grupos de medidas formais que servem para orientar os gestores de usabilidade sobre a qualidade em uso de um sistema. As métricas são validadas e avaliadas a partir de estudos de caso que se baseiam no desempenho dos usuários. Através de métricas descreve-se, por exemplo, o quão rápido um usuário pode executar uma tarefa, o número de erros cometidos em uma tarefa, o tempo de aprendizagem, e as avaliações subjetivas dos usuários (Usability First, 2010).

Portanto, a métrica pode estar relacionada ao desempenho do usuário sobre um determinado

conjunto de tarefas de teste. As medidas mais básicas baseiam-se na definição de usabilidade como uma métrica de qualidade (NIELSEN, 2001).

As métricas estão agrupadas como é exibido na tabela 1.

	GRUPOS DE MÉTRICAS DE USABILIDADE	
	Métrica	Medida
APRENDIZADO	Tempo para treinamento	O total de horas recomendadas para que um usuário aprenda a utilizar o sistema
EFICÁCIA	Percentual de sucesso	A razão entre a quantidade de tarefas concluídas e o total de tarefas alocadas
	Grau de operacionalidade	A razão entre os comandos utilizáveis e os disponíveis
	Percentual de falhas	A razão entre a quantidade de tarefas perdidas e o total de tarefas alocadas
EFICIÊNCIA	Tempo médio de conclusão	Tempo necessário para completar um grupo de tarefas
	Disponibilidade de comandos	Quantidade de alternativas ou atalhos para um mesmo comando
	Tempo de ociosidade	Tempo que um usuário consome para continuar a tarefa
SATISFAÇÃO	Utilidade de um produto ou serviço	Nível de alinhamento das tarefas com as reais necessidades dos usuários
	Número de reclamações	Quantidade de vezes que os usuários reclamaram do serviço
	Percentual de aceitação	Índice de aprovação do sistema pelos usuários
SEGURANÇA	Tolerância a falhas	Percentual de falhas que podem ser recuperadas

É fácil especificar as métricas de usabilidade, porém é difícil coletá-las. Pois, estas métricas poderão variar conforme o contexto de uso e muitas vezes tendem à subjetividade. Além de serem difíceis de coletar, as métricas encarecem os testes de usabilidade, pois a alocação de tempo e recursos é elevada. Porém é possível diminuir estes aspectos negativos apenas escolhendo os métodos apropriados de avaliação.

Fica a dica

A usabilidade tornou-se tão importante que existe o Dia Mundial da Usabilidade. Veja mais em: http://www.diamundialdausabilidade.com.br/2011/.

A usabilidade é tão importante que o Governo Federal do Brasil criou a Cartilha de Usabilidade. Veja mais em: http://www.governoeletronico.gov.br/acoes-e-projetos/padroes-brasil-e-gov/cartilha-de-usabilidade/.

E não perca tempo! Conheça o Instituto Brasileiro de Amigabilidade e Usabilidade IBRAU acessando o site: http://www.ibrau.com.br/.

Recordando

O Teste de Usabilidade garante que o sistema é fácil de operar e possui ergonomia, tema tão importante que empresas como a Nokia, Apple, Motorola, entre outras, possuem investimentos e setores para testes de usabilidade.

8 - PLANEJAMENTO DE AVALIAÇÃO

8.1 – INTRODUÇÃO

As avaliações de usabilidade deverão ser práticas realizadas várias vezes durante todo o processo de desenvolvimento de Software e antes da entrega do produto final. O resultado será um sistema de melhor qualidade e de maior compreensão dos usuários para o qual o sistema é projetado. Entre os elementos básicos de uma avaliação de usabilidade, estão (RUBIN, 2008):

- Desenvolvimento de questionários e testes de objetivos ao invés de hipóteses.
- Uso de um exemplo representativo de usuário final que poderá ou não ser aleatoriamente escolhido.
- Representação do ambiente de trabalho atual.
- Observação de usuários finais que usam ou revisam uma representação do produto.
- Entrevista controlada ou eventualmente extensiva e sondagem dos participantes pelo moderador de testes e recomendações de melhorias no projeto do produto.

É evidente que a forma de melhorar os processos de produção de Software é incorporar métodos de avaliação como testes de usabilidade através de processos iterativos. Os testes de usabilidade caracterizam-se por uma abordagem baseada no "feedback" do usuário, ou seja, suas respostas durante cada ação. Este processo envolve usuários finais que determinam o grau de satisfação relacionado a uma aplicação. É importante fazer o bom uso desta ferramenta para detectar e corrigir alguns defeitos, pois estes geralmente são imperceptíveis pelas equipes de desenvolvimento.

Para facilitar a avaliação de um Software e então melhorar a detecção de erros de usabilidade, as equipes de testes de Software desenvolveram uma classificação de métodos de avaliação. Estes métodos dividem-se em inspeções e testes. A inspeção é realizada sem a presença de usuários finais, enquanto o teste tem a participação de destes usuários no processo de avaliação.

As avaliações podem utilizar dois métodos e isso vai depender do tempo, custo e recursos que se planeja utilizar no teste, desta forma é importante estar atento a todas as necessidades.

Normalmente, qualquer teste inicia pelo plano de teste, que aponta os recursos, objetivos do teste e o que será testado.

Desta forma, é possível preparar o ambiente de teste para podermos iniciar a aplicação dos testes, sempre lembrando que o objetivo do teste é buscar defeito.

O plano de teste de usabilidade pode seguir os mesmos passos, mas a execução possui algumas particularidades, dependendo do método a ser utilizado. Entre as tarefas podemos destacar:

- Análise das tarefas do usuário.
- Avaliação heurística.
- Comparação do tempo para execução de tarefas, etc.

Fica a dica!

A avaliação de usabilidade é tão importante que diversos pesquisadores tratam sobre o assunto. Veja mais nos links abaixo:

- http://www.labceo.com.br/bibliografia/archive/files/h-13_fb714b0be0.pdf
- http://www.devmedia.com.br/artigo-engenharia-de-software-5-avaliacao-de-usabilidade/10247

Recordando

O Teste de Usabilidade garante que o sistema é fácil de operar e possui ergonomia, mas para saber isto é necessário executar uma avaliação.

9 - MÉTODOS DE INSPEÇÃO

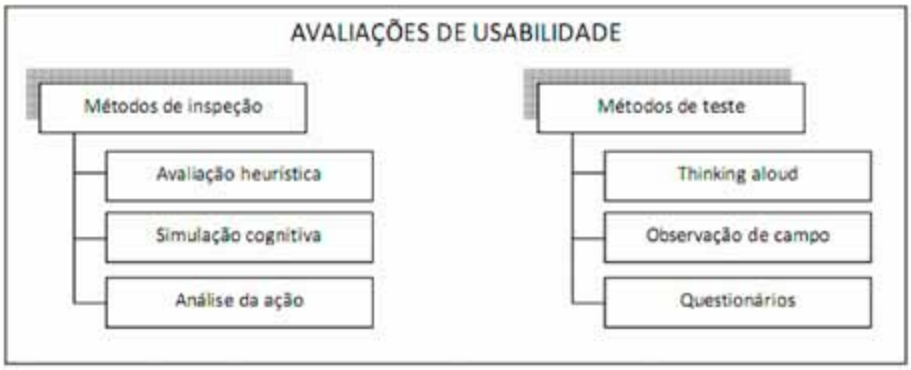
9.1 – INTRODUÇÃO

A inspeção é um conjunto de métodos que serve para identificar problemas de uma interface. É uma avaliação de conformidade, ou seja, se um produto está de acordo com as normas e padrões estabelecidos.

Segundo NIELSEN (1992) "a avaliação heurística é um método de inspeção que compreende um conjunto de examinadores que avaliam a interface do usuário e que através de princípios de usabilidade tentam encontrar problemas possíveis".

Este método é tido como eficaz, pois os especialistas afirmam que até 75% dos problemas podem ser resolvidos através da avaliação heurística. Para garantir sua eficácia é necessário um mínimo de três avaliadores. Esta quantidade varia com a complexidade do sistema.

Baseando-se em pesquisas com centenas de usuários, Jakob Nielsen elaborou dez princípios de usabilidade que devem ser seguidos por todos os desenvolvedores de interfaces.



1) O "feedback" determina que o usuário deverá estar sempre informado sobre a situação e as respostas apropriadas do sistema.
2) O sistema deverá falar a linguagem do usuário, ou seja, os diálogos deverão ser utilizados de forma clara utilizando-se palavras e conceitos conhecidos pelo usuário.
3) As saídas deverão estar claramente demarcadas, pois os usuários geralmente escolhem funções por engano e eles podem abortá-las no momento oportuno.
4) A consistência é a garantia para que uma mesma ação sempre tenha um resultado previsível.
5) Prevenir erros contra situações atípicas é melhor contra situações desagradáveis.
6) O sistema deverá minimizar a sobrecarga de memória do usuário, ou seja, ter instruções visíveis para as funções que diminuam a necessidade de memorização de comandos.
7) Os atalhos são considerados os aceleradores de comandos e podem ser vistos como práticos por todo tipo de usuário.
8) Os diálogos deverão ser simples e naturais. Evitar informações irrelevantes ou que raramente serão solicitadas.
9) Boas mensagens de erro significam que o sistema deverá indicar qual foi o problema e sugerir uma possível solução para este.
10) A ajuda e a documentação deverão estar disponíveis e facilmente acessíveis.

A simulação cognitiva ou percurso cognitivo faz uma análise detalhada da sequência de ações do usuário. A sequência de ações refere-se às etapas necessárias para que um indivíduo conclua uma tarefa. O foco deste tipo de avaliação é saber o quanto um sistema é fácil de aprender. Para alcançar este objetivo, não basta ler um manual de instruções

ou fazer treinamento específico para os usuários do sistema, até porque muitos usuários preferem aprender a utilizar um sistema apenas explorando suas funcionalidades. Então os avaliadores realizam verificações passo a passo das tarefas e, no fim de cada inspeção fornecem um relatório com justificativas sobre a avaliação de cada etapa.

Na equipe de avaliadores, cada participante assume um papel específico, como moderar o teste, registrar as reações do usuário, ou até mesmo assumir o papel do próprio usuário do sistema. Enquanto os passos são executados, um membro da equipe registra as dificuldades e dúvidas encontradas. Os requisitos para se fazer uma simulação cognitiva são: uma descrição do sistema, a descrição da tarefa que um usuário deve executar no sistema e lista completa das ações necessárias para completar a tarefa.

A análise da ação é similar ao percurso cognitivo e requer uma inspeção mais controlada da sequência de ações que um usuário realiza para completar uma tarefa. Esta análise envolve a decomposição de tarefas em ações individuais, por exemplo, mover o cursor até o menu ou clicar em um botão. Para cada ação é calculado um tempo necessário para conclusão. Esta estimativa de tempo torna a análise da ação muito complexa mesmo quando uma tarefa simples é estudada e a representação das ações só pode ser interpretada por quem conduziu a análise.

10 - MÉTODOS DE TESTE DE USABILIDADE

10.1 – INTRODUÇÃO

Os testes com os usuários finais são os principais métodos de avaliação de usabilidade. A importância destes métodos deve-se aos dados reais obtidos dos usuários sobre como estes utilizam o sistema e quais problemas poderão encontrar durante a execução de suas tarefas diárias.

Uma forma de avaliar a usabilidade através de usuários finais é estimular que estes declarem o que estão pensando enquanto utilizam a interface. Quando há qualquer problema de interface que atrapalhe uma ação, o usuário expõe suas dificuldades que serão registradas por um observador. O método de verbalizar os pensamentos dos usuários é denominado de "thinking aloud" (do inglês para "pensando alto"). A iniciativa de entender como os usuários interpretam um sistema facilita o mapeamento de partes deste que causam problemas de usabilidade. Para complementar este tipo de teste, os especialistas podem fazer uso de técnicas, como o mouse-tracking que mostra em quais regiões da tela o ponteiro do mouse passou com maior frequência.

Outra forma de avaliar os usuários é a observação ou estudo de campo. É uma pesquisa etnográfica, ou seja, envolve uma investigação dos usuários em seu local de trabalho de como estes utilizam o sistema através de entrevistas e observações. A observação direta poderá ser importuna para o usuário, causando uma alteração em seu comportamento em relação ao sistema. É por esta razão que o estudo de campo deverá interferir o mínimo possível nas tarefas dos usuários, caso contrário implicará em análises equivocadas.

Muitos aspectos de usabilidade podem ser melhorados por meio de consultas indiretas aos usuários. Por exemplo, os questionários são úteis para indicar desde o nível de satisfação dos usuários até sugestões de melhorias do sistema. As opiniões dos usuários são subjetivas, e por esta razão, os questionários devem abranger um maior número de usuários. Não basta coletar opiniões, elas devem convergir para algum problema que deverá ser corrigido. Um grande desafio é interpretar as verdadeiras necessidades dos usuários através desta técnica informal. É por esta razão que os usuários deverão ser experientes quanto ao sistema avaliado.

A tabela 2 define um quadro comparativo entre as várias metodologias de avaliação de usabilidade abordadas.

	MÉTODOS DE INSPEÇÃO			MÉTODOS DE TESTE		
	Avaliação Heurística	Percurso Cognitivo	Análise de Ação	"Thinking Aloud"	Observação de campo	Questionários
Etapas	todas	todas	projeto	projeto	testes finais	todas
Consumo de tempo	baixo	médio	alto	alto	médio	baixo
Número de usuários	nenhum	nenhum	nenhum	3 ou mais	20 ou mais	30 ou mais
Número de avaliadores	3 ou mais	3 ou mais	1 a 2	1	1 ou mais	1
Necessidade de equipamentos	baixa	baixa	baixa	alta	média	baixa
Necessidade de experiência	médio	alto	alto	médio	alto	baixo
É intrusivo?	não	não	não	sim	sim	não

Note que os métodos possuem diferenças significativas. É importante, ao realizar o planejamento de teste de usabilidade, saber os recursos de pessoal e de custo necessários para a realização dos testes.

Obviamente, é possível realizar tanto a inspeção quanto o teste, mas lembre-se que o teste não pode ser mais caro do que o produto, caso contrário, não teremos o retorno de investimento adequado ao sistema.

11 - MÉTRICAS DE QUALIDADE

11.1 – INTRODUÇÃO

A partir da medição das características do produto e da eficiência dos processos de produção é possível avaliar a qualidade dos mesmos e iniciar o processo contínuo de melhoria da qualidade.

A relação com a qualidade é suficiente para considerarmos as métricas como imprescindíveis. Alguns autores citam que alguns projetos obtiveram sucesso sem o uso de métricas. Essa afirmação gera algumas questões:

1) Qual foi o critério utilizado para o sucesso?

Satisfação do cliente ou uma meta/padrão pré-estabelecido.

2) Como saber se a satisfação do cliente ou uma meta/padrão pré-estabelecida foi alcançada?

Através do uso de algum tipo de avaliação ou comparação, considerando-se que a comparação não deixa de ser um método de avaliação do grau de similaridade. No restante desse documento será adotado apenas o termo avaliação.

3) Como avaliar ou comparar?

Utilizando algum tipo de métrica.

Ao avaliar, nos baseamos em métricas coletadas durante ou ao final da produção. Exemplos:

- Os testes do ciclo atual devem encontrar menos defeitos que o anterior.
- Os testes devem garantir cobertura de no mínimo 80% das funcionalidades.

As avaliações podem estar relacionadas com:

- **Produtos:** avalia-se a qualidade.
- **Processos:** avalia-se diretamente a produtividade e indiretamente a qualidade.

As métricas podem ser:

- Subjetivas: obtidas através de opiniões especializadas, análise sensorial, satisfação do usuário, etc.
- Objetivas: obtidas através de instrumentos de medição, contagens, etc.

Portanto, esse estudo considera que não é possível determinar o sucesso de um projeto sem avaliação e que para avaliar é necessário o uso de métricas, dentro do contexto de que um projeto de sucesso é aquele que atinge os resultados esperados segundo o planejado.

As métricas são necessárias para:

- Analisar qualidade e produtividade do processo de desenvolvimento e manutenção, bem como do produto de Software construído.
- Qualificar a performance técnica dos produtos do ponto de vista do desenvolvedor.
- Medidas funcionais são necessárias para qualificar a performance dos produtos pela perspectiva do usuário.
- Devem ser independentes das decisões do desenvolvimento técnico e

implementação.
- Utilizadas para comparar a produtividade de diferentes técnicas e tecnologias.

A partir do exposto, seguem informações mais específicas sobre métricas dentro do contexto de testes de Software.

11.2 - USO DE MÉTRICAS

Para avaliar os processos e produtos é necessário coletar informações, enfim, medir as características que irão subsidiar a avaliação. Com base na avaliação é possível melhorar a produtividade e a qualidade, ou simplesmente ratificar um estágio temporariamente satisfatório. Entende-se por características qualquer atributo do processo ou produto.

1) O que medir?

Todas as características determinantes do "produto", segundo os requisitos, devem ser medidas, desde que estejam relacionadas à produtividade e/ou à qualidade. Não se deve estabelecer métricas se a contribuição que as mesmas terão na avaliação da produtividade e/ou qualidade não for relevante.

2) Como e quando medir os "produtos"?

Através da coleta de métricas relacionadas aos requisitos do produto. As medições devem acontecer nas várias fases do "produto", de forma que possibilitem a identificação, o mais cedo possível, de eventuais deficiências do mesmo.

3) Como e quando medir os processos?

As medições devem ser incorporadas aos processos, coletadas durante sua execução, preferencialmente de forma automatizada. Deve-se evitar a criação de processos exclusivos de medição e, caso seja necessário criá-los, é fundamental que essa medição agregue alto valor, afinal exigirá um esforço adicional.

Estabelecidas as métricas, como e quando serão coletadas, resta ainda definir:

- Quem irá coletá-las.
- Quais ferramentas técnicas e práticas serão utilizadas.
- Com que frequência serão coletadas.
- Onde os dados serão armazenados.
- Quem fará a avaliação.

Os responsáveis pelas coletas devem estar treinados para a atividade, que deve ser monitorada.

Isoladamente, as métricas não oferecem benefícios. Porém, elas estão vinculadas à avaliação e consequentes decisões de continuísmo ou mudança que visem à melhoria da qualidade.

11.3 - RAZÕES PARA SE MEDIR O SOFTWARE

- Indtficativa de aquisição de novas ferramentas ou de treinamentos adicionais.

As métricas de Software, do ponto de vista de medição, podem ser divididas em duas categorias: medidas diretas e indiretas.

MEDIDAS DIRETAS	MEDIDAS INDIRETAS

• Custo • Esforço • Linhas de Código • Velocidade de Execução • Memória • Nº de Erros	• Funcionalidade • Qualidade • Complexidade • Eficiência • Confiabilidade • Manutenibilidade

Fica a dica

Este assunto é de extrema importância, sendo pesquisado por Doutores no mundo todo. Segue abaixo alguns links importantes:

- http://www.riopomba.ifsudestemg.edu.br/dcc/dcc/materiais/1022789570_Qualidade%20de%20Software.pdf
- http://www.testexpert.com.br/?q=node/669
- http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ESI2009_2/Aula03_04.pdf

Recordando

As métricas de qualidade nos dão a base do que é necessário para termos qualidade. É com base nas métricas que podemos realizar os testes e dar garantia da qualidade de um produto.

12 - TMMI e CMMI

12.1 – INTRODUÇÃO AO CMMI

O CMMI é um Framework, um conjunto de métodos comprovados que servem para garantir a maturidade de desenvolvimento de Software em uma empresa. O Framework foi criado para priorizar a melhoria, de forma a garantir o alinhamento entre as atividades de melhoria de processo e os objetivos de negócio da organização.

O CMMI é uma compilação de boas práticas divididas em 22 áreas de processos, criado em 1987 e mantido pelo SEI (Software Engeneering Institute) da Universidade Carnegie Mellon e patrocinado pelo Departamento de Defesa dos Estados Unidos da América, sendo que a última revisão do CMMI foi a 1.3, lançada em 2010.

As 22 áreas de processos são divididas em 4 PA´s, e possui uma escala de 5 níveis de maturidade.

As PA ou AP (Área de Processo) são agrupamentos de práticas comuns de uma determinada disciplina.

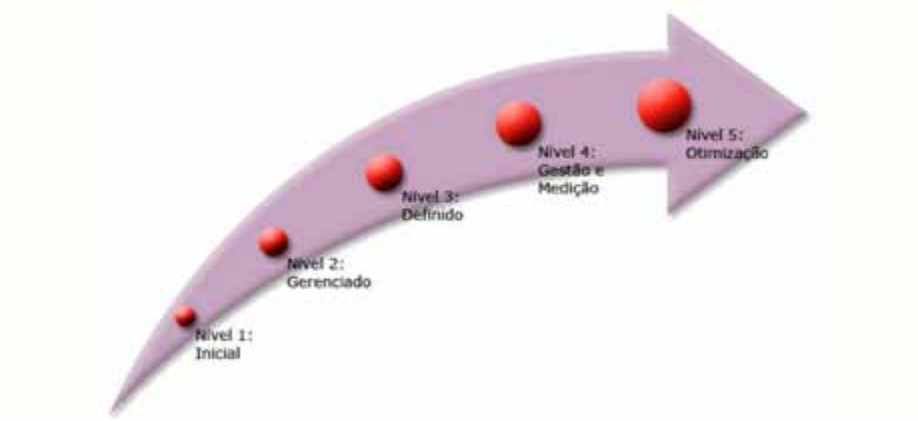
Categoria	Área de Processo
Engenharia	• Integração do Produto • Desenvolvimento de Requisitos • Gestão de Requisitos • Solução Técnica • Validação • Verificação
Gestão de Processos	• Inovação e Implementação Organizacional • Definição do Processo Organizacional • Enfoque no Processo Organizacional • Formação Organizacional

Gestão de Projetos	• Gestão Integrada do Projeto • Monitoramento e Controle de Projeto • Gestão Quantitativa do Projeto • Planejamento de Projeto • Gestão de Risco • Gestão de Acordo com o Fornecedor
Suporte	• Análise e Resolução Causal • Gestão de Configurações • Análise das Decisões e Resolução • Medição e Análise • Garantia da Qualidade do Processo e do Produto

12.2 - INTRODUÇÃO AO TMMI

O TMMI é um Framework baseado no CMMI e possui os mesmos objetivos, que é garantir a maturidade do Software, mas o enfoque é o Processo de Teste.

O TMMI é mantido pela TMMI Foundation que criou todo o processo, sempre seguindo as bases do CMMI, o resultado é um Framework completo que possui os mesmos níveis do CMMI, mas totalmente focado em teste.



Os níveis do TMMI são 5 assim como no CMMI, como pode ser visto na figura acima, seguindo a mesma ideia.

O TMMI deriva alguns recursos do ISTQB Standart Glossary que possui um enfoque maior em teste possuindo inclusive certificação para Teste de Software.

O objetivo do TMMI é levar ao teste as boas práticas da indústria como a melhoria contínua.

- **Nível 1:** é o caos que reina, ou seja, não há processos, não há definição real de como realizar os testes, não há padrões nem como fazer.
- **Nível 2:** inicia o processo gerencial possuindo políticas e estratégias de teste. O planejamento dos testes já é realizado, há monitoramento e controle durante as realizações dos testes e há um ambiente para os testes.
- **Nível 3:** já temos um processo bem definido, além dos recursos do nível 2 temos uma organização dos testes, uma integração contínua, um ciclo de vida dos testes bem definido e testes não funcionais, como testes unitários e de performance.
- **Nível 4:** temos um processo de medição de qualidade em conjunto com o processo, utilizando-se de análise e resultados reais dos testes com o objetivo de garantia de qualidade, aplicando revisão contínua e melhorias.
- **Nível 5:** que é o mais alto de todos, é o nível otimizado, em que o processo

de teste já foi otimizado e faz parte do dia a dia da empresa, para isso temos prevenção aos defeitos, utilizando TDD, por exemplo. O processo de teste passa a ser otimizado e automatizado e há controle de qualidade contínua nos processos de teste.

Notemos que tanto o CMMI como o TMMI são semelhantes, possuindo pequenas diferenças entre eles. Estas diferenças estão principalmente no fato de que o foco de cada um é diferente, enquanto um se foca no processo de desenvolvimento completo, o outro se foca no processo de teste.

Fica a dica!

Tanto o CMMI como o TMMI são importantes para os processos de qualidade de Software, para saber mais acesse:

TMMI Framework

http://www.tmmifoundation.org/downloads/tmmi/TMMi%20Framework.pdf

TMMI Foundation

http://www.tmmifoundation.org

CMMI Framework

http://www.sei.cmu.edu/library/abstracts/news-at-sei/featuresept98pdf.cfm

GLOSSÁRIO

- **CMMI:** Capatibility Maturity Model Integration ou podemos dizer Modelo Integrado de Capacidade para Maturidade.
- **TMMI:** Test Maturity Model Integration ou Modelo Integrado de Maturidade em Teste.

Referências

Apache JMeter. Disponível em: jmeter.apache.org/. Acessado em: 22/06/2012.

Artigo Engenharia de Software 5 - Avaliação de Usabilidade. Disponível em: http://www.devmedia.com.br/artigo-engenharia-de-software-5-avaliacao-de-usabilidade/10247. Acessado em: 25/06/2012.

BERKUN, Scott. *A arte do gerenciamento de projetos.* Porto Alegre: Bookman, 2008.

Cartilha de Usabilidade. Disponível em: http://www.governoeletronico.gov.br/acoes-e--projetos/padroes-brasil-e-gov/cartilha-de-usabilidade/. Acessado em: 25/06/2012.

CMM Integration (CMMI) Framework. Disponível em: http://www.sei.cmu.edu/library/abstracts/news-at-sei/featuresept98pdf.cfm. Acessado em: 25/06/2012.

DAMAS, Luís. *SQL: Structured Query Language.* 6. Ed. atual. e ampl. Rio de Janeiro: LTC, 2007. x, 384 p.

Diretrizes: Teste Unitário. Disponível em: http://www.wthreex.com/rup/process/modguide/md_untst.htm. Acessado em: 20/06/2012.

Ciclo de Deming. Disponível em: http://sandrocan.files.wordpress.com/2009/01/00361.jpg. Acessado em: 20/06/2012.

IBRAU – Instituto Brasileiro de Ambiguidade e Usabilidade. Disponível em: http://www.ibrau.com.br/. Acessado em: 25/06/2012.

KERZNER, Harold. *Gestão de projetos: as melhores práticas.* 2° ed. Porto Alegre: Bookman, 2006.

Padrões de Qualidade de Software e Métricas de Software. Disponível em: http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ESI2009_2/Aula03_04.pdf. Acessado em: 25/06/2012.

PHPUnit Manual. Disponível em: http://www.phpunit.de/manual/3.4/en/index.html. Acessado em: 22/06/2012.

PRESSMAN. Roger S. *Engenharia de Software*. São Paulo. 6º ed. Porto Alegre: MacGraw--Hill, 2006.

Qualidade de Software com PHP. Disponível em: www.fgrweb.com.br/wp. Acessado em: 22/06/2012.

Qualidade de Software. Disponível em: http://www.riopomba.ifsudestemg.edu.br/dcc/dcc/materiais/1022789570_Qualidade%20de%20Software.pdf. Acessado em: 25/06/2012.

Qualidade, Qualidade de Software e Garantia da Qualidade de Software são as mesmas coisas? Disponível em: http://www.testexpert.com.br/?q=node/669. Acessado em: 25/06/2012.

SimpleTest. Acessado em: simpletest.org/. Disponível em: 22/06/2012.

Softwares de Gerenciamento de Projeto: Uma Avaliação da Usabilidade. Disponível em: http://www.labceo.com.br/bibliografia/archive/files/h-13_fb714b0be0.pdf. Acessado em: 25/06/2012.

SOMMERVILLE. Ian. Engenharia de Software. 6º ed. Pearson Education Companion, 2003.
Test Maturity Model Integration (TMMI). Disponível em: http://www.tmmifoundation.org/downloads/tmmi/TMMi%20Framework.pdf. Acessado em: 25/06/2012.

TMMI Foundation. Disponível em: http://www.tmmifoundation.org. Acessado em: 25/06/2012.

Usando o JMeter para teste de performance. Disponível em: http://fgrweb.com.br/wp/?p=68. Acessado em: 22/06/2012.

XAVIER, Carlos Magno da Silva; VIVACQUA, Flavio Ribeiro; MACEDO, Otualp Sarmento de; XAVIER, Luis Fernando da Silva. *Metodologia de Gerenciamento de Projetos – Methodware.* Rio de Janeiro: Editora Brasport, 2009.

alfamacursos.com.br